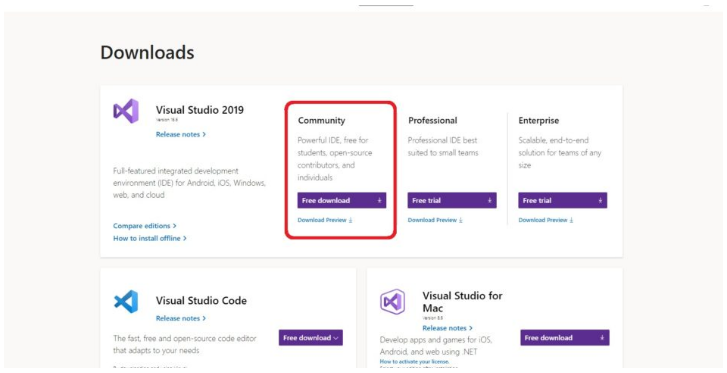
(https://www.computervision.zone)

# Face Recognition Opencv + Attendance Project

In this tutorial we are going to learn how to perform Face recognition using opencv with high accuracy. We will first briefly go through the theory and learn the basic implementation. Then we will create an Attendance project that will use webcam to detect faces and record the attendance live in an excel sheet. Face recognition is a topic that is very popular both among beginners and experienced computer vision majors. This is because it is something that is quiet useful in tons of applications.

## Installations

The installation process for this project is a bit more than usual. First we have to download a C++ compiler. We can do this by installing Visual Studios. You can download the community version for free from their website (https://visualstudio.microsoft.com/downloads/). Once the intaller we will run it and select the 'Desktop development with C++'. The download and installation will take some time as it is a few Gbs.



Downloads

Visual Studio 2019 — Full-featured integrated development environment (IDE) for Android, iOS, Windows, web, and cloud

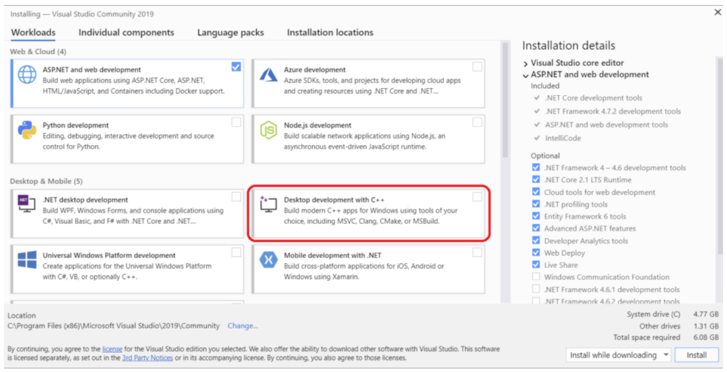
Community — Powerful IDE, free for students, open-source contributors, and individuals

Professional — Professional IDE best suited to small teams

Enterprise — Scalable, end-to-end solution for teams of any size

Visual Studio Code — The fast, free and open-source code editor that adapts to your needs

Visual Studio for Mac — Develop apps and games for iOS, Android, and web using .NET



Installing — Visual Studio Community 2019

Web & Cloud (4)

- ASP.NET and web development — Build web applications using ASP.NET Core, ASP.NET, HTML/JavaScript, and Containers including Docker support. [✓]
- Azure development — Azure SDKs, tools, and projects for developing cloud apps and creating resources using .NET Core and .NET...
- Python development — Editing, debugging, interactive development and source control for Python.
- Node.js development — Build scalable network applications using Node.js, an asynchronous event-driven JavaScript runtime.

Desktop & Mobile (5)

- .NET desktop development — Build WPF, Windows Forms, and console applications using C#, Visual Basic, and F# with .NET Core and .NET...
- Desktop development with C++ — Build modern C++ apps for Windows using tools of your choice, including MSVC, Clang, CMake, or MSBuild.
- Universal Windows Platform development — Create applications for the Universal Windows Platform with C#, VB, or optionally C++.
- Mobile development with .NET — Build cross-platform applications for iOS, Android or Windows using Xamarin.

Installation details
- Visual Studio core editor
- ASP.NET and web development
  - Included: ✓ .NET Core development tools; ✓ .NET Framework 4.7.2 development tools; ✓ ASP.NET and web development tools; ✓ IntelliCode
  - Optional: ✓ .NET Framework 4 – 4.6 development tools; ✓ .NET Core 2.1 LTS Runtime; ✓ Cloud tools for web development; ✓ .NET profiling tools; ✓ Entity Framework 6 tools; ✓ Advanced ASP.NET features; ✓ Developer Analytics tools; ✓ Web Deploy; ✓ Live Share; ☐ Windows Communication Foundation; ☐ .NET Framework 4.6.1 development tools; ☐ .NET Framework 4.6.2 development tools

Location C:\Program Files (x86)\Microsoft Visual Studio\2019\Community

System drive (C) 4.77 GB
Other drives 1.31 GB
Total space required 6.08 GB

By continuing, you agree to the license for the Visual Studio edition you selected. We also offer the ability to download other software with Visual Studio. This software is licensed separately, as set out in the 3rd Party Notices or in its accompanying license. By continuing, you also agree to those licenses.

After completing and restarting the computer, now we will head on to our Pycharm project. Here we will install the required packages. Below is the list.

- cmake
- dlib
- face_recognition
- numpy
- opencv-python

## Understanding the problem

Although many face recognition opencv algorithms have been developed over the years, their speed and accuracy balance has not been quiet optimal . But some recent advancements have shown promise. A good example is Facebook, where they are able to tag you and your friends with just a few images of training and with accuracy as high as 98%. So how does this work . Today we will try to replicate similar results using a face recognition library developed by Adam Geitgey. Lets look at the 4 problems he explained in his article (https://medium.com/@ageitgey/machine-learning-is-fun-part-4-modern-face-recognition-with-deep-learning-c3cffc121d78).

Face recognition is a series of several problems:

1. First, look at a picture and find all the faces in it
2. Second, focus on each face and be able to understand that even if a face is turned in a weird direction or in bad lighting, it is still the same person.
3. Third, be able to pick out unique features of the face that you can use to tell it apart from other people— like how big the eyes are, how long the face is, etc.
4. Finally, compare the unique features of that face to all the people you already know to determine the person's name.

## Face Recognition

### Importing

First we will import the relevant libraries

```
import face_recognition
import cv2
import numpy as np
```

The Process of Recognition can be divided into 3 steps.

## Step 1

### Loading Images and Converting to RGB.

The Face Recognition package consists of a load image function that loads the image. Once imported the image has to be converted to RGB.

```
imgElon = face_recognition.load_image_file('ImagesBasic/Elon Musk.jpg')
imgElon = cv2.cvtColor(imgElon,cv2.COLOR_BGR2RGB)
imgTest = face_recognition.load_image_file('ImagesBasic/Elon Test.jpg')
imgTest = cv2.cvtColor(imgTest,cv2.COLOR_BGR2RGB)
```

Elon Musk

Elon Musk Test

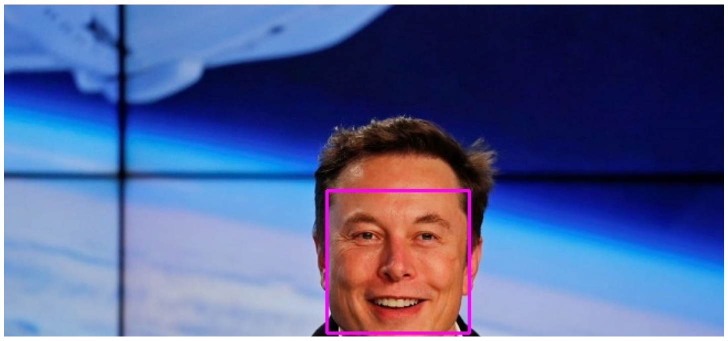
## Step 2

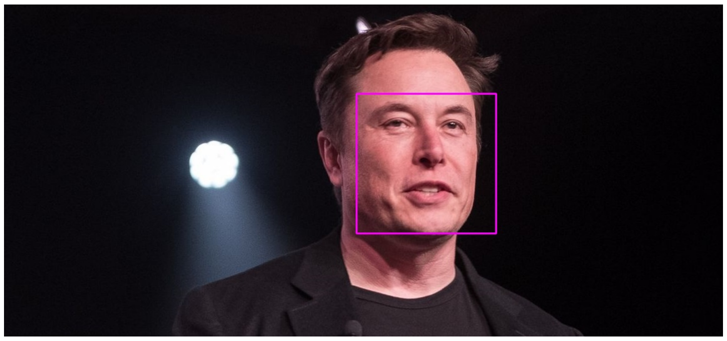
### Find Faces Locations and Encodings

In the second step we will use the true functionality of the face recognition library. First we will find the faces in our images . This is done using HOG (Histogram of Oriented Gradients) at the backend. Once we have the face they are warped to remove unwanted rotations. Then the image is feed to a pretrained neural network that out puts 128 measurements that are unique to that particular face. The parts that the model measures is not known as this is what the model learns by itself when it was trained. Lucky for us all this is done is just 2 lines of code. Once we have the face locations and the encodings we can draw rectangles around our faces.

```
faceLoc = face_recognition.face_locations(imgElon)[0]
encodeElon = face_recognition.face_encodings(imgElon)[0]
cv2.rectangle(imgElon,(faceLoc[3],faceLoc[0]),(faceLoc[1],faceLoc[2]),(255,0,255),2) # top, right, bottom, left

faceLocTest = face_recognition.face_locations(imgTest)[0]
encodeTest = face_recognition.face_encodings(imgTest)[0]
cv2.rectangle(imgTest,(faceLocTest[3],faceLocTest[0]),(faceLocTest[1],faceLocTest[2]),(255,0,255),2)
```





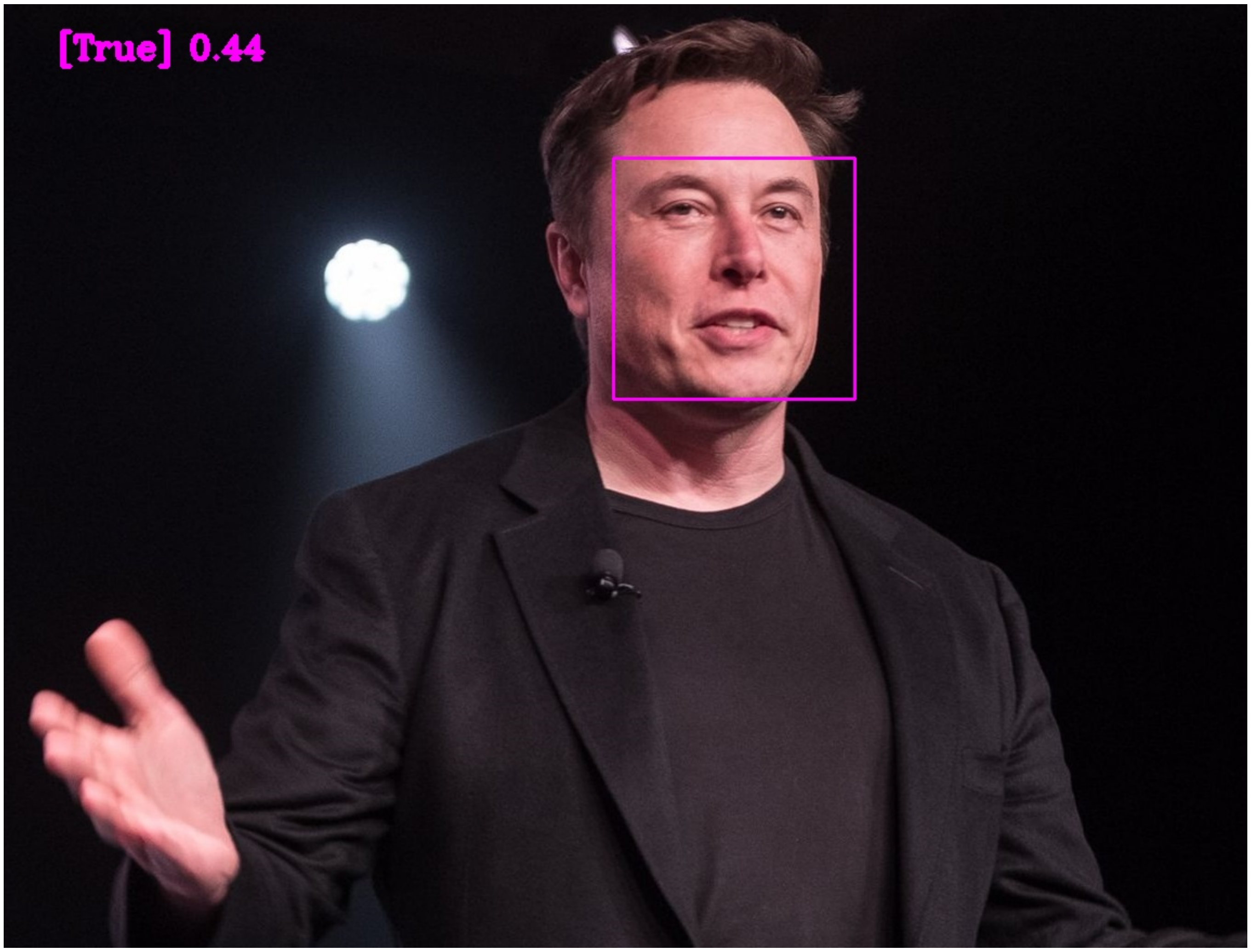
## Step 3

### Compare Faces and Find Distance

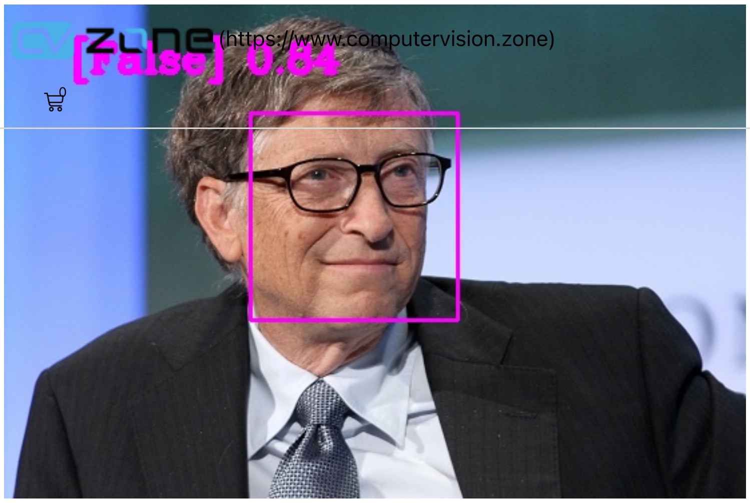
Once we have the encodings for both faces, then we can compare these 128 measurements of these two faces to find similarities. To compare the package uses one of the most common machine Learning methods linear SVM classifier. We can use the compare_faces function to find if the faces match. This function returns True or False. Similarly we can use the face_distance function to find how likely is the faces match in terms of numbers. This is helpul particularly when there are multiple faces to detect from.

```
results = face_recognition.compare_faces([encodeElon], encodeTest)
faceDis = face_recognition.face_distance([encodeElon], encodeTest)
cv2.putText(imgTest,f'{results} {round(faceDis[0],2)} ',(50,50),cv2.FONT_HERSHEY_COMPLEX,1,(255,0,255),3)
```

If we run this with the test image we get the value True, indicating that the face found is of Elon Musk. The is distance between the faces is 0.44. The lower the distance the better the match.



Let try it with another testing image. This time we will test it with an image of Bill Gates. As we can see the result is False and the distance is much higher than before indicating a bad match.



(https://www.computervision.zone)

## Attendance Project

Now using the methods we have seen above, we will develop an attendance system where the user is automatically logged when they are detected in the camera. We will store the name along with the time when they appeared.

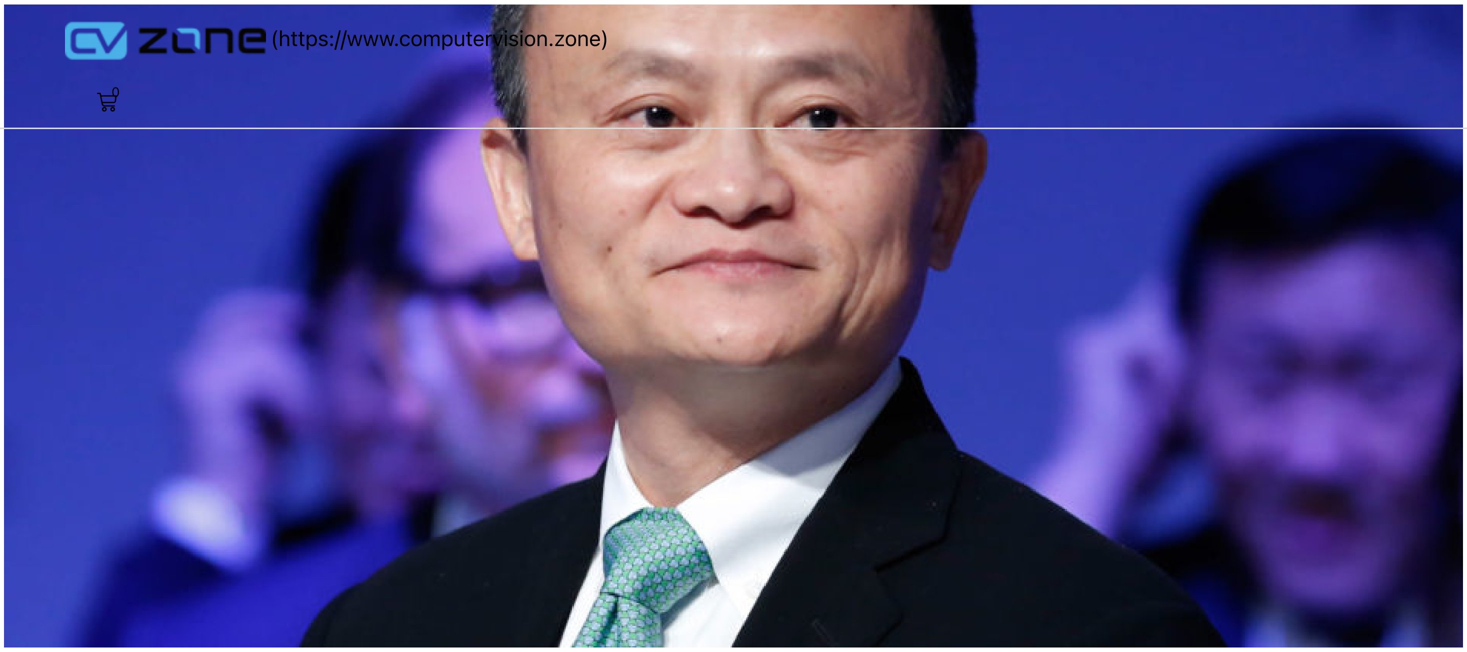
### Importing Images

As we have imported before we can use the same face_recognition.load_image_file() function to import our images. But when we have multiple images, importing them individually can become messy. Therefore we will write a script to import all images in a given folder at once. For this we will need the os library so we will import that first. We will store all the images in one list and their names in another.

```
import face_recognition
import cv2
import numpy as np
import os
```

```
path = 'ImagesAttendance'
images = []     # LIST CONTAINING ALL THE IMAGES
className = []    # LIST CONTAINING ALL THE CORRESPONDING CLASS Names
myList = os.listdir(path)
print("Total Classes Detected:",len(myList))
for x,cl in enumerate(myList):
        curImg = cv2.imread(f'{path}/{cl}')
        images.append(curImg)
        className.append(os.path.splitext(cl)[0])
```

(https://www.computervision.zone)



## Compute Encodings

Now that we have a list of images we can iterate through those and create a corresponding encoded list for known faces. To do this we will create a function. As earlier we will first convert it into RGB and then find its encoding using the face_encodings() function. Then we will append each encoding to our list.

```
def findEncodings(images):
    encodeList = []
    for img in images:
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        encode = face_recognition.face_encodings(img)[0]
        encodeList.append(encode)
    return encodeList
```

Now we can simply call this function with the images list as the input arguments.

```
encodeListKnown = findEncodings(images)
print('Encodings Complete')
```

## The While loop

The while loop is created to run the webcam. But before the while loop we have to create a video capture object so that we can grab frames from the webcam.

```
cap = cv2.VideoCapture(0)
```

### Webcam Image

First we will read the image from the webcam and then resize it to quarter the size. This is done to increase the speed of the system. Even though the image being used is 1/4 th of the original, we will still use the original size while displaying. Next we will convert it to RGB.

```
while True:
    success, img = cap.read()
    imgS = cv2.resize(img, (0, 0), fx=0.25, fy=0.25)
    imgS = cv2.cvtColor(imgS, cv2.COLOR_BGR2RGB)
```

### Webcam Encodings

Once we have the webcam frame we will find all the faces in our image. The face_locations function is used for this purpose. Later we will find the face_encodings as well.

```
    facesCurFrame = face_recognition.face_locations(imgS)
    encodesCurFrame = face_recognition.face_encodings(imgS, facesCurFrame)
```

## Find Matches

(https://www.computervision.zone)

Now we can match the current face encodings to our known faces encoding list to find the matches. We will also compute the distance. This is done to find the best match in case more than one face is detected at a time.

```
    for encodeFace,faceLoc in zip(encodesCurFrame,facesCurFrame):
        matches = face_recognition.compare_faces(encodeListKnown, encodeFace)
        faceDis = face_recognition.face_distance(encodeListKnown, encodeFace)
```

Once we have the list of face distances we can find the minimum one, as this would be the best match.

```
        matchIndex = np.argmin(faceDis)
```

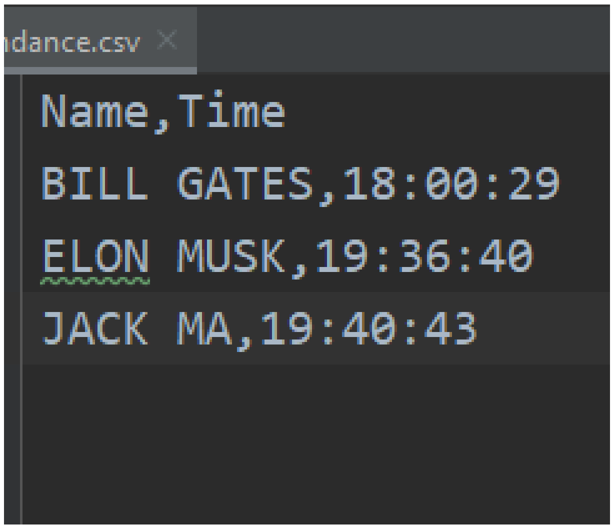
Now based on the index value we can determine the name of the person and display it on the original Image.

```
        if matches[matchIndex]:
            name = className[matchIndex].upper()
            y1,x2,y2,x1=faceLoc
            y1, x2, y2, x1 = y1*4,x2*4,y2*4,x1*4
            cv2.rectangle(img, (x1, y1), (x2, y2), (0, 255, 0), 2)
            cv2.rectangle(img, (x1, y2 - 35), (x2, y2), (0, 255, 0), cv2.FILLED)
            cv2.putText(img, name, (x1 + 6, y2 - 6), cv2.FONT_HERSHEY_DUPLEX, 1.0, (255, 255, 255), 1)
```

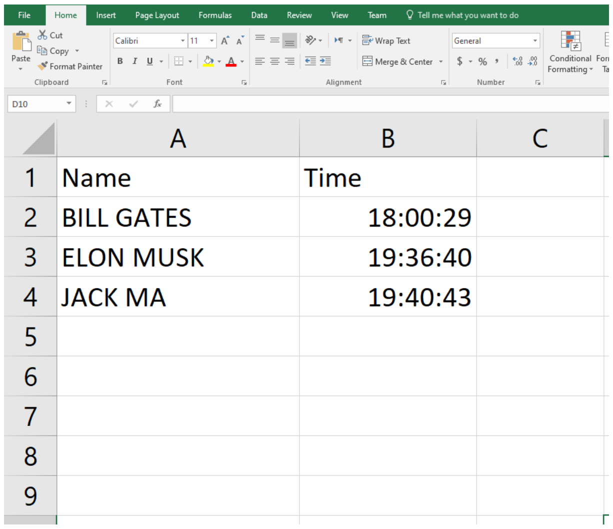
## Marking Attendance

Lastly we are going to add the automated attendance code. We will start by writing a function that requires only one input which is the name of the user. First we open our Attendance file which is in csv format. Then we read all the lines and iterate through each line using a for loop. Next we can split using comma ','. This will allow us to get the first element which is the name of the user. If the user in the camera already has an entry in the file then nothing will happen. On the other hand if the user is new then the name of the user along with the current time stamp will be stored. We can use the datetime class in the date time package to get the current time.

```
def markAttendance(name):
    with open('Attendance.csv','r+') as f:
        myDataList = f.readlines()
        nameList =[]
        for line in myDataList:
            entry = line.split(',')
            nameList.append(entry[0])
        if name not in  line:
            now = datetime.now()
            dt_string = now.strftime("%H:%M:%S")
            f.writelines(f'\n{name},{dt_string}')
```



```
Name,Time
BILL GATES,18:00:29
ELON MUSK,19:36:40
JACK MA,19:40:43
```



| | A | B | C |
|---|---|---|---|
| 1 | Name | Time | |
| 2 | BILL GATES | 18:00:29 | |
| 3 | ELON MUSK | 19:36:40 | |
| 4 | JACK MA | 19:40:43 | |
| 5 | | | |
| 6 | | | |
| 7 | | | |
| 8 | | | |
| 9 | | | |

## Labeling Unknown faces as well

To find the unknown faces we will replace

```
        if matches[matchIndex]:
            name = classNames[matchIndex].upper()
            #print(name)
            y1,x2,y2,x1 = faceLoc
            y1, x2, y2, x1 = y1*4,x2*4,y2*4,x1*4
            cv2.rectangle(img,(x1,y1),(x2,y2),(0,255,0),2)
            cv2.rectangle(img,(x1,y2-35),(x2,y2),(0,255,0),cv2.FILLED)
            cv2.putText(img,name,(x1+6,y2-6),cv2.FONT_HERSHEY_COMPLEX,1,(255,255,255),2)
            markAttendance(name)
```

with this

```
        if faceDis[matchIndex]< 0.50:
            name = classNames[matchIndex].upper()
            markAttendance(name)
        else: name = 'Unknown'
        #print(name)
        y1,x2,y2,x1 = faceLoc
        y1, x2, y2, x1 = y1*4,x2*4,y2*4,x1*4
        cv2.rectangle(img,(x1,y1),(x2,y2),(0,255,0),2)
        cv2.rectangle(img,(x1,y2-35),(x2,y2),(0,255,0),cv2.FILLED)
        cv2.putText(img,name,(x1+6,y2-6),cv2.FONT_HERSHEY_COMPLEX,1,(255,255,255),2)
```

All this does is to check if the distance to our min face is less than 0.5 or not. If its not then this means the person is unknown so we change the name to unknown and don't mark the attendance.

So in conclusion the face recognition opencv method is one of the easiest and fastest ways to implement the facial recognition.

## Video Tutorial

## Complete Code

You can access the full code through enrolling on this free course.

## Share

**[DISPLAY_ULTIMATE_SOCIAL_ICONS]**

## 3 Responses

**Snehal Chodankar** says: Jul 4, 2020 at 3:00 am (https://www.computervision.zone/face-recognition-opencv/#comment-149)

(https://www.computervision.zone)

Sir, Can we limit the number of faces detected in the webcam...?
I tried this code but every time the webcam turns on, and starts detecting faces, it lags a lot...
So I just wanted to know if it's possible to limit the faces detected...

Sir, even if u provide me the logic then it would be a great help...

Reply

↪ **Murtaza Hassan (https://www.computervision.zone/user/murtazahassan/)** says: Jul 5, 2020 at 3:43 am (https://www.computervision.zone/face-recognition-opencv/#comment-154)

The lag is bcz the computation is heavy.

Reply

↪ **Snehal Chodankar** says: Jul 18, 2020 at 2:42 pm (https://www.computervision.zone/face-recognition-opencv/#comment-226)

So can we limit the faces detected per frame...?

If we do so, computation will be reduced to sufficient extent I feel...

Reply

## Leave a Reply

Logged in as MacDrug Escobar (https://www.computervision.zone/wp-admin/profile.php). Log out? (https://www.computervision.zone/wp-login.php?action=logout&redirect_to=https%3A%2F%2Fwww.computervision.zone%2Fface-recognition-opencv%2F&_wpnonce=bfb1a618df)

Comment

Post Comment

*Terms of Use (https://www.computervision.zone/terms/)*

*Pirvacy Policy (https://www.computervision.zone/privacy-policy/)*

### Contact

Email (mailto:contact@computervision.zone)